IN THE CLAIMS:

The following listing of claims will replace all prior listings of claims in the application:

1. (Currently Amended): A method for synchronizing divergent graphics samples included in a group of data samples processed in a programmable graphics processing unit, the method comprising:

executing each instruction of an instruction sequence simultaneously on each sample of the group;

determining that a divergence has occurred between a subset of the samples in the group, whereby a call/return operation is executed on the subset of the samples of the group;

detecting that a first sample of ~~a group~~ the subset of samples has encountered a first synch token;

determining whether any of the other samples of the ~~group~~ subset has encountered a synch token; ~~and~~

determining whether the synch token encountered by any of the other samples of the ~~group~~ subset is the first synch token;and[[.]]

synchronizing the subset of samples with the other samples of the group for processing a next instruction in the instruction sequence only if all the samples of the subset have encountered the first synch token.

2. (Currently Amended): The method of claim 1, further comprising the step of ~~determining whether to initiate~~ initiating a time out when the first sample of the subset encounters the first synch token; and

processing the next instruction in the instruction sequence only if all the samples of the subset encounter the first synch token within a defined time period.

3. (Cancelled)

4. (Cancelled)

5. (Cancelled)

6. (Original): The method of claim 1, further comprising the step of initiating termination steps if the synch token encountered by any of the other samples in the group is not the first synch token.

7. (Cancelled)

8. (Currently Amended): The method of claim [[1]]2, further comprising the step of holding the first sample idle once the first sample has encountered the first synch token until the time out period elapses.

9. (Original): The method of claim 1, wherein determining that a divergence has occurred comprises determining that a first program counter of a plurality of program counters is different than a second program counter of the plurality of program counters, each program counter of the plurality of program counters corresponding to a different one of the samples of the group of samples.

10. (Original): The method of claim 9, wherein the first program counter being different than the second program counter results from a conditional branch or a jump.

11. (Original): The method of claim 1, wherein determining that a divergence has occurred comprises determining that a first subroutine depth of a plurality of subroutine depths is different than a second subroutine depth of the plurality of subroutine depths, each subroutine depth of the plurality of subroutine depths corresponding to a different one of the samples of the group of samples.

12. (Original): The method of claim 11, wherein the first subroutine depth being different than the second subroutine depth results from a call-return.

13. (Currently Amended): A method for processing divergent graphics samples in a programmable graphics processing unit, the method comprising:

processing samples of a group of samples in non-divergent mode;

determining whether each program counter of a plurality of program counters is the same, each program counter of the plurality of program counters corresponding to a different one of the samples of the group of samples; ~~and~~

determining whether each subroutine depth of a plurality of subroutine depths is the same, each subroutine depth of the plurality of subroutine depths corresponding to a different one of the samples of the group of samples;[[.]]

processing each of the other samples in the group of samples in non-divergent mode, after processing the one or more divergent samples.

14. (Original): The method of claim 13, further comprising the step of processing one or more divergent samples through a remainder of a program if a first program counter of the plurality of program counters is different than a second program counter of the plurality of program counters.

15. (Original): The method of claim 14, wherein the first program counter being different than the second program counter results from a conditional branch or a jump.

16. (Cancelled)

17. (Cancelled)

18. (Original): The method of claim 13, wherein the first subroutine depth being different than the second subroutine depth relates to a call-return.

19. (Cancelled)

20. (Currently Amended): A system for synchronizing divergent graphics samples in a programmable graphics processing unit, the system comprising:

a plurality of processing threads, each processing thread corresponding to a different sample of a group of samples and configured to contain a program counter, a subroutine depth and state data; and

a plurality of stacks, each stack corresponding to a different sample of the group of samples and configured to store state data in one or more stack levels, wherein a first portion of each stack resides in a dedicated local storage resource and a second portion of each stack resides in local memory.

21. (Original): The system of claim 20, wherein the subroutine depth of a first sample is equal to the number of the one or more stack levels of a first stack that contain state data, the first stack corresponding to the first sample.

22. (Original): The system of claim 20, wherein each stack resides in a dedicated local storage resource.

23. (Original): The system of claim 20, wherein a first portion of each stack resides in a dedicated local storage resource and a second portion of each stack resides in local memory.

24. (Original): A system for synchronizing divergent graphics samples included in a group of data samples processed in a programmable graphics processing unit, the system comprising:

means for simultaneously executing each instruction of an instruction sequence on each sample of the group of samples,

means for determining that a divergence has occurred between a subset of the samples in the group, whereby a call/return operation is executed on the subset of the samples of the group;

means for detecting that a first sample of a ~~group~~ subset of samples has encountered a first synch token;

means for determining whether each of the other samples of the ~~group~~ subset has encountered a synch token; ~~and~~

means for determining whether the synch token encountered by each of the other samples in the ~~group~~ subset is the first synch token[[.]] ; and

means for synchronizing the subset of samples with the other samples of the group for processing a next instruction in the instruction sequence only if all the samples of the subset have encountered the first synch token.

25. (Original): The system of claim 24, further comprising means for processing the group of samples in non-divergent mode if the synch token encountered by each of the other samples in the group is the first synch token.

26. (New): The method of claim 1, wherein determining whether divergence has occurred includes corresponding the program counters (PC) of threads assigned to the samples, and if the samples of the group do not all have the same PC then divergence has occurred, and if the samples of the group all have the same PC then synchronous processing continues.

27. (New): The method of claim 1, including holding idle the samples of the group on which the call/return is not executed.